

The Grid File I/O (Gfio) Library

Version 1.0

ROB LUCCHESI AND ARLINDO DA SILVA

Data Assimilation Office, NASA/GSFC, Greenbelt, MD 20771

November 2, 1998 (Original design October 1997)

Contents

1 System Overview	2
2 Routine/Function Prologues	4
2.1 GFIO_Create – Creates a DAO gridded file for writing (Source File: gfio.F)	4
2.2 GFIO_Open – Opens an existing DAO gridded file (Source File: gfio.F) . . .	14
2.3 GFIO_PutVar – Write a variable to the file (Source File: gfio.F)	15
2.4 GFIO_GetVar – Read a variable from the file (Source File: gfio.F)	23
2.5 GFIO_DimInquire – Gets dimension information from a GFIO file. (Source File: gfio.F)	29
2.6 GFIO_Inquire – Get information about a GFIO file. (Source File: gfio.F) . .	31
2.7 GFIO_Close – Closes file (Source File: gfio.F)	39
2.8 GFIO_PutIntAtt – Write a user-defined integer attribute (Source File: gfio.F)	40
2.9 GFIO_PutRealAtt – Write a user-defined real attribute (Source File: gfio.F)	42
2.10 GFIO_PutCharAtt – Write a user-defined character attribute (Source File: gfio.F)	44
2.11 GFIO_GetAttNames – Get global attribute names (Source File: gfio.F) . . .	45
2.12 GFIO_AttInquire – Get information about an attribute (Source File: gfio.F)	47
2.13 GFIO.GetIntAtt – Read a user-defined integer attribute (Source File: gfio.F)	49
2.14 GFIO.GetRealAtt – Read a user-defined real attribute (Source File: gfio.F)	51
2.15 GFIO.GetCharAtt – Read a user-defined character attribute (Source File: gfio.F)	53

1 System Overview

Basic Requirements:

-
- (1) Design an interface for writing HDF format data files from the June 1998 GEOS-3 production system without requiring the direct insertion of HDF Toolkit calls in the code.
 - (2) Design this interface to be flexible enough to support usage in other DAO applications that read or write HDF data.
 - (3) Output files should conform the COARDS conventions. This allows the data to be immediately usable by GrADS, other visualization packages and utilites such as ncdump.
 - (4) Provide a library that is callable from a Fortran 77 application with a portable interface.
 - (5) The library must also be callable by C, perhaps with the use of a tool like Cfortran.h.

The primary motivation behind GFIO is to provide an easy way for the GEOS-DAS to write HDF format data while hiding calls to the HDF libraries. Additionally, it is hoped that this library will be of general use for reading or writing HDF files in applications other than the GEOS-DAS.

The typical calling sequence for creating a file would be:

```
GFIO_Create(...)  
  
GFIO_PutVar(...)  
GFIO_PutVar(...)  
GFIO_PutVar(...)  
. . .  
GFIO_Close(...)
```

One could subsequently open the file for more writing with:

```
GFIO_Open(...)
```

NOTES:

- * Surface data is permitted in the same file as upper air data, however all upper air data must be defined with the same number of levels but it is not necessary to write data for each defined level. In the case that data is not written for a given level, HDF will put fills.
- * Packing is not yet implemented.
- * The time increment cannot be defined as 0, even if only writing one time.
- * Files are written using the NetCDF interface provided in the HDF library. The files conform to the COARDS conventions, meaning they have specific metadata defined by the convention.

2 Routine/Function Prologues

2.1 GFIO_Create – Creates a DAO gridded file for writing (Source File: *gfio.F*)

This routine is used to open a new file for a GFIO stream. Packing is not yet supported. Information about each opened stream is stored in a COMMON block contained in *gfio.h*. This information is later used by GFIO_PutVar. GFIO_OpenW should be used to open an existing file for writing.

INTERFACE:

```
subroutine GFIO_Create ( fname, title, source, contact, amiss,
&                                im, jm, km, lon, lat, levs, levunits,
&                                yyyyymmdd_beg, hhmmss_beg, timinc,
&                                nvars, vname, vtitle, vunits, kmvar,
&                                valid_range, packing_range, prec,
&                                fid, rc )
```

USES:

```
Implicit NONE
#include "netcdf.inc"
#include "gfio.h"
```

INPUT PARAMETERS:

		! ----- Global Metadata -----
character(*)	fname	! File name
character(*)	title	! A title for the data set
character(*)	source	! Source of data, e.g. NASA/DAO
character(*)	contact	! Who to contact about the data set, e.g., ! 'Contact data@dao.gsfc.nasa.gov'
real	amiss	! Missing value such as 1.0E15
		! ----- Dimension Metadata -----
integer	im	! size of longitudinal dimension
integer	jm	! size of latitudinal dimension
integer	km	! size of vertical dimension ! (surface only=1)
real	lon(im)	! longitude of center of gridbox in ! degrees east of Greenwich (can be ! -180 -> 180 or 0 -> 360)
real	lat(jm)	! latitude of center of gridbox in ! degrees north of equator
real	levs(km)	! Level (units given by levunits) of ! center of gridbox
character(*)	levunits	! units of level dimension, e.g., ! "millibar", "hPa", or "sigma_level"
integer	yyyyymmdd_beg	! First year-month-day to be written
integer	hhmmss_beg	! First hour-minute-second to be written

```

integer      timinc      ! Increment between output times (HHMMSS)

                                ! ----- Variable Metadata -----
integer      nvars       ! number of variables in file
character(*) vname(nvars) ! variable short name, e.g., "hght"
character(*) vtitle(nvars)! variable long name, e.g.,
                           ! "Geopotential Height"
character(*) vunits(nvars)! variable units, e.g., "meter/second"
integer      kmvar(nvars)! number of levels for variable; it can
                           ! either be 0 (2-D fields) or equal to km

real       valid_range(2,nvars) ! Variable valid range; GFIO_PutVar
                                ! will return a non-fatal error if a value is
                                ! outside of this range. IMPORTANT: If packing
                                ! is not desired for a given variable, YOU MUST
                                ! set both components of valid_range to amiss.
                                ! ----- Packing Metadata -----
real       packing_range(2,nvars) ! Packing range to be used for 16-bit packing
                                ! of each variable. IMPORTANT: If packing is not
                                ! desired for a given variable, YOU MUST set both
                                ! components of packing_range to amiss.
                                ! NOTE:
                                ! * The packing algorithm sets all values
                                !   outside the packing range to missing.
                                ! * The larger the packing range, the greater
                                !   the loss of precision.
integer      prec        ! Desired precision of data:
                           ! 0 = 32 bit
                           ! 1 = 64 bit
                           ! NOTE: mixing precision in the same
                           ! * Mixing 32 and 64 bit precision in the
                           !   same file is not supported.
                           ! * If packing is turned on for a variable,
                           !   the prec flag is ignored.

```

OUTPUT PARAMETERS:

```

integer      fid        ! File handle
integer      rc         ! Error return code:
                           ! rc = 0  all is well
                           ! rc = -1 time increment is 0
                           ! NetCDF Errors
                           !
                           ! -----
                           ! rc = -30  error from nccre (file create)
                           ! rc = -31  error from ncddef
                           ! rc = -32  error from ncvdef (dimension variable)
                           ! rc = -33  error from ncaptc (dimension attribute)
                           ! rc = -34  error from ncvdef (variable)

```

```

      !  rc = -35  error from ncaptc (variable attribute)
      !  rc = -36  error from ncaptc/ncapt (global attribute)
      !  rc = -37  error from ncendf
      !  rc = -38  error from ncvpt (dimension variable)

```

REVISION HISTORY:

1997.09.13	da Silva/Lucchesi	Initial interface design.
1997.09.22	Lucchesi	Added timinc to interface.
1998.02.10	Lucchesi	Added support for applications running with 64-bit reals.
1998.02.17	Lucchesi	Added time_inc, begin_time, and begin_date attributes to the time dimension.
1998.03.30	Lucchesi	Documentation expanded. Clean-up of code.
1998.07.07	Lucchesi	Removed vids from argument list
1998.07.09	Lucchesi	Converted timinc to seconds before saving
1998.10.09	Lucchesi	Precision flag, documentation changes.
1998.10.27	Lucchesi	Added support for packing and range checks

CONTENTS:

```
! REAL*4 variables for 32-bit output to netCDF file.
```

```

real*4 amiss_32
real*4 lon_32(im), lat_32(jm), levs_32(km)
real*4 scale_32, offset_32
real*4 high_32,low_32
integer vid(nvars)
integer i, j
integer timeid, latid, lonid, levid
integer timedim, latdim, londim, levdim
integer dims3D(4), dims2D(3)
integer corner(4), edges(4)
character*80 timeUnits
logical :: surfaceOnly = .TRUE.
character*8 strBuf
character*14 dateString
integer year,mon,day,hour,min,sec
integer err
integer incSecs

```

```
! Variables for packing
```

```

integer*2 amiss_16
real*4 pRange_32(2,nvars),vRange_32(2,nvars)

```

```

logical packflag

! Set metadata strings. These metadata values are specified in the
! COARDS conventions

character (len=50) :: lonName = "longitude"
character (len=50) :: lonUnits = "degrees_east"
character (len=50) :: latName = "latitude"
character (len=50) :: latUnits = "degrees_north"
character (len=50) :: levName = "vertical level"
c           levUnits: specified by user in argument list
character (len=50) :: timeName = "time"
c           timeUnits: string is built below
character (len=50) :: conventions = "COARDS"
character (len=50) :: history = "File written by GFIO v1.0"

amiss_16 = PACK_FILL

! Basic error-checking.

if (timinc .eq. 0) then
  rc=-1
  return
endif

! Check to see if there is only surface data in this file definition

do i=1,nvars
  if (kmvar(i) .NE. 0) then
    surfaceOnly = .FALSE.
    exit
  endif
enddo

! Convert double-precision output variables to single-precision

do i=1,im
  lon_32(i) = lon(i)
enddo
do i=1,jm
  lat_32(i) = lat(i)
enddo
do i=1,km
  levs_32(i) = levs(i)
enddo
do j=1,nvars
  do i=1,2
    vRange_32(i,j) = valid_range(i,j)
  enddo
enddo

```

```
        pRange_32(i,j) = packing_range(i,j)
    enddo
enddo

amiss_32 = amiss

! Convert time increment to seconds.

      write (strBuf,203) timinc
203  format (I6)
      read (strBuf,204) hour, min, sec
204  format (3I2)
      incSecs = hour*3600 + min*60 + sec

! Make NetCDF errors non-fatal, but issue warning messages.

      call ncpopt(NCVERBOS)

! Create the new NetCDF file. [ Enter define mode. ]

      fid = nccre (fname, NCCLLOB, rc)
      if (err("Create: can't open file",rc,-30) .LT. 0) return

! Define dimensions.

      londim = ncddef (fid, 'lon', im, rc)
      if (err("Create: error defining lon",rc,-31) .LT. 0) return
      latdim = ncddef (fid, 'lat', jm, rc)
      if (err("Create: error defining lat",rc,-31) .LT. 0) return
      if (.NOT. surfaceOnly) then
          levdim = ncddef (fid, 'lev', km, rc)
          if (err("Create: error defining lev",rc,-31) .LT. 0) return
      endif
      timedim = ncddef(fid, 'time', NCUNLIM, rc)
      if (err("Create: error defining time",rc,-31) .LT. 0) return

! Define dimension variables.

      lonid = ncvdef (fid, 'lon', NCFLOAT, 1, londim, rc)
      if (err("Create: error creating lon",rc,-32) .LT. 0) return
      latid = ncvdef (fid, 'lat', NCFLOAT, 1, latdim, rc)
      if (err("Create: error creating lat",rc,-32) .LT. 0) return
      if (.NOT. surfaceOnly) then
          levid = ncvdef (fid, 'lev', NCFLOAT, 1, levdim, rc)
          if (err("Create: error creating lev",rc,-32) .LT. 0) return
      endif
      timeid = ncvdef (fid, 'time', NCLONG, 1, timedim, rc)
      if (err("Create: error creating time",rc,-32) .LT. 0) return
```

```
! Set attributes for dimensions.

call ncaptc (fid,lonid,'long_name',NCCHAR,LEN_TRIM(lonName),
.           lonName,rc)
if (err("Create: error creating lon attribute",rc,-33) .LT. 0)
.   return
call ncaptc (fid,lonid,'units',NCCHAR,LEN_TRIM(lonUnits),
.           lonUnits,rc)
if (err("Create: error creating lon attribute",rc,-33) .LT. 0)
.   return

call ncaptc (fid,latid,'long_name',NCCHAR,LEN_TRIM(latName),
.           latName,rc)
if (err("Create: error creating lat attribute",rc,-33) .LT. 0)
.   return
call ncaptc (fid,latid,'units',NCCHAR,LEN_TRIM(latUnits),
.           latUnits,rc)
if (err("Create: error creating lat attribute",rc,-33) .LT. 0)
.   return

if (.NOT. surfaceOnly) then
  call ncaptc (fid,levid,'long_name',NCCHAR,LEN_TRIM(levName),
.           levName,rc)
  if (err("Create: error creating lev attribute",rc,-33) .LT. 0)
.   return
  call ncaptc (fid,levid,'units',NCCHAR,LEN_TRIM(levunits),
.           levunits,rc)
  if (err("Create: error creating lev attribute",rc,-33) .LT. 0)
.   return
endif

call ncaptc (fid, timeid, 'long_name', NCCHAR, LEN_TRIM(timeName),
.           timeName, rc)
if (err("Create: error creating time attribute",rc,-33) .LT. 0)
.   return

write (dateString,200) yyyyymmdd_beg, hhmmss_beg
200 format (I8,I6)
read (dateString,201) year,mon,day,hour,min,sec
201 format (I4,5I2)
write (timeUnits,202) year,mon,day,hour,min,sec
202 format ('minutes since ',I4.4,'-',I2.2,'-',I2.2,' ',I2.2,':',
.           I2.2,':',I2.2)
call ncaptc (fid, timeid, 'units', NCCHAR, LEN_TRIM(timeUnits),
.           timeUnits, rc)
if (err("Create: error creating time attribute",rc,-33) .LT. 0)
.   return
```

```

call ncapt (fid, timeid, 'time_increment', NCLONG, 1, incSecs,rc)
if (err("Create: error creating time attribute",rc,-33) .LT. 0)
.  return
call ncapt (fid,timeid,'begin_date',NCLONG,1,yyyymmdd_beg,rc)
if (err("Create: error creating time attribute",rc,-33) .LT. 0)
.  return
call ncapt (fid,timeid,'begin_time',NCLONG,1,hhmmss_beg,rc)
if (err("Create: error creating time attribute",rc,-33) .LT. 0)
.  return

dims3D(4) = timedim
dims3D(3) = levdim
dims3D(2) = latdim
dims3D(1) = londim

dims2D(3) = timedim
dims2D(2) = latdim
dims2D(1) = londim

scale_32 = 1.0      ! No packing for now.
offset_32 = 0.0      ! No packing for now.

! Set up packing attributes for each variable.
! Define physical variables.  Set attributes for physical variables.

do i=1,nvars
  if (pRange_32(1,i) .NE. amiss_32 .OR. pRange_32(2,i) .NE.
. amiss_32) then
    packflag = .TRUE.
  else
    packflag = .FALSE.
  endif
  if ( kmvar(i) .eq. 0 ) then
    if (packflag) then
      vid(i) = ncvdef (fid, vname(i), NCSHORT, 3, dims2D, rc)
    else if (prec .EQ. 1) then
      vid(i) = ncvdef (fid, vname(i), NCDOUBLE, 3, dims2D, rc)
    else
      vid(i) = ncvdef (fid, vname(i), NCFLOAT, 3, dims2D, rc)
    endif
  else
    if (packflag) then
      vid(i) = ncvdef (fid, vname(i), NCSHORT, 4, dims3D, rc)
    else if (prec .EQ. 1) then
      vid(i) = ncvdef (fid, vname(i), NCDOUBLE, 4, dims3D, rc)
    else
      vid(i) = ncvdef (fid, vname(i), NCFLOAT, 4, dims3D, rc)
    endif
  endif
enddo

```

```

        endif
        if (err("Create: error defining variable",rc,-34) .LT. 0)
        .    return

        call ncaptc (fid, vid(i), 'long_name', NCCHAR,
        .            LEN_TRIM(vtitle(i)),vtitle(i), rc)
        if (err("Create: error defining variable attribute",rc,-35)
        .       .LT. 0) return
        call ncaptc (fid, vid(i), 'units', NCCHAR,
        .            LEN_TRIM(vunits(i)),vunits(i), rc)
        if (err("Create: error defining variable attribute",rc,-35)
        .       .LT. 0) return

!
! Set up packing info.
!

        if (packflag) then
            if (pRange_32(1,i) .GT. pRange_32(2,i)) then
                high_32 = pRange_32(1,i)
                low_32 = pRange_32(2,i)
            else
                high_32 = pRange_32(2,i)
                low_32 = pRange_32(1,i)
            endif
            scale_32 = (high_32 - low_32)/PACK_BITS*2
            offset_32 = high_32 - scale_32*PACK_BITS
            if (scale_32 .EQ. 0.0) then           ! If packing range is 0,
                scale_32 = 1.0                   ! default to no packing.
                offset_32 = 0.0
            endif
            call ncapt (fid,vid(i),'scale_factor',NCFLOAT,1,scale_32,rc)
            if (err("Create: error defining variable attribute",rc,-35)
            .       .LT. 0) return
            call ncapt (fid,vid(i),'add_offset',NCFLOAT,1,offset_32,rc)
            if (err("Create: error defining variable attribute",rc,-35)
            .       .LT. 0) return
            call ncapt (fid,vid(i),'packmin',NCFLOAT,1,low_32,rc)
            if (err("Create: error defining variable attribute",rc,-35)
            .       .LT. 0) return
            call ncapt (fid,vid(i),'packmax',NCFLOAT,1,high_32,rc)
            if (err("Create: error defining variable attribute",rc,-35)
            .       .LT. 0) return
            call ncapt (fid,vid(i),'missing_value',NCSHORT,1,amiss_16,rc)
            if (err("Create: error defining variable attribute",rc,-35)
            .       .LT. 0) return
            call ncapt (fid,vid(i),'fmissing_value',NCFLOAT,1,amiss_32,rc)
            if (err("Create: error defining variable attribute",rc,-35)

```

```

.     .LT. 0) return
else
scale_32 = 1.0      ! No packing.
offset_32 = 0.0      ! No packing.
call ncapt (fid,vid(i),'scale_factor',NCFLOAT,1,scale_32,rc)
if (err("Create: error defining variable attribute",rc,-35)
.     .LT. 0) return
call ncapt (fid,vid(i),'add_offset',NCFLOAT,1,offset_32,rc)
if (err("Create: error defining variable attribute",rc,-35)
.     .LT. 0) return
call ncapt (fid,vid(i),'missing_value',NCFLOAT,1,amiss_32,rc)
if (err("Create: error defining variable attribute",rc,-35)
.     .LT. 0) return
call ncapt (fid,vid(i),'fmissing_value',NCFLOAT,1,amiss_32,rc)
if (err("Create: error defining variable attribute",rc,-35)
.     .LT. 0) return

endif
if (vRange_32(1,i) .NE. amiss_32 .OR. vRange_32(2,i) .NE.
.     amiss_32) then
if (vRange_32(1,i) .GT. vRange_32(2,i)) then
high_32 = vRange_32(1,i)
low_32  = vRange_32(2,i)
else
high_32 = vRange_32(2,i)
low_32  = vRange_32(1,i)
endif
call ncapt (fid,vid(i),'vmin',NCFLOAT,1,low_32,rc)
if (err("Create: error defining variable attribute",rc,-35)
.     .LT. 0) return
call ncapt (fid,vid(i),'vmax',NCFLOAT,1,high_32,rc)
if (err("Create: error defining variable attribute",rc,-35)
.     .LT. 0) return
else
call ncapt (fid,vid(i),'vmin',NCFLOAT,1,amiss_32,rc)
if (err("Create: error defining variable attribute",rc,-35)
.     .LT. 0) return
call ncapt (fid,vid(i),'vmax',NCFLOAT,1,amiss_32,rc)
if (err("Create: error defining variable attribute",rc,-35)
.     .LT. 0) return

endif
enddo

! Define global file attributes.

call ncaptc (fid,NCGLOBAL,'Conventions',NCCHAR,
.           LEN_TRIM(conventions),conventions,rc)

```

```

if (err("Create: error defining Conventions",rc,-36).LT. 0)
.  return
call ncaptc (fid,NCGLOBAL,'Source',NCCHAR,LEN_TRIM(source),
.           source,rc)
if (err("Create: error defining Source",rc,-36).LT. 0) return
call ncaptc (fid,NCGLOBAL,'Title',NCCHAR,LEN_TRIM(title),title,
.           rc)
if (err("Create: error defining Title",rc,-36).LT. 0) return
call ncaptc (fid,NCGLOBAL,'Contact',NCCHAR,LEN_TRIM(contact),
.           contact,rc)
if (err("Create: error defining Contact",rc,-36).LT. 0) return
call ncaptc (fid,NCGLOBAL,'History',NCCHAR,LEN_TRIM(history),
.           history,rc)
if (err("Create: error defining History",rc,-36).LT. 0) return

! Exit define mode.

call ncendf (fid, rc)
if (err("Create: error exiting define mode",rc,-37) .LT. 0)
.  return

! Write out dimension variables.

corner(1) = 1
edges(1) = im
call ncvpt (fid, lonid, corner, edges, lon_32, rc)
if (err("Create: error writing lons",rc,-38) .LT. 0) return

corner(1) = 1
edges(1) = jm
call ncvpt (fid, latid, corner, edges, lat_32, rc)
if (err("Create: error writing lats",rc,-38) .LT. 0) return

if (.NOT. surfaceOnly) then
  corner(1) = 1
  edges(1) = km
  call ncvpt (fid, levid, corner, edges, levs_32, rc)
  if (err("Create: error writing levs",rc,-38) .LT. 0) return
endif

corner(1) = 1
edges(1) = 1
call ncvpt (fid, timeid, corner, edges, 0, rc)
if (err("Create: error writing times",rc,-38) .LT. 0) return

rc=0
return
end

```

```
!-----  
!      NASA/GSFC, Data Assimilation Office, Code 910.3, GEOS/DAS      !  
!-----
```

2.2 GFIO_Open – Opens an existing DAO gridded file (Source File: *gfio.F*)

This routine opens an existing DAO gridded file. The file mode will be read/write. If the application already knows the contents of the file, it may begin interaction with the file using the returned file handle. Otherwise, the file handle can be used with the "inquire" routines to gather information about the contents. A negative return code indicates there were problems opening the file.

INTERFACE:

```
subroutine GFIO_Open ( fname, fmode, fid, rc )
```

USES:

```
Implicit NONE
#include "netcdf.inc"
#include "gfio.h"
```

INPUT PARAMETERS:

character(*)	fname	! File name
integer	fmode	! File mode: ! 0 for READ-WRITE ! non-zero for READ-ONLY

OUTPUT PARAMETERS:

integer	fid	! File handle
integer	rc	! Error return code: ! rc = 0 All is well ! rc = -39 error from ncopen (file open)

REVISION HISTORY:

1998.07.02	Lucchesi	Initial interface design.
1998.07.07	Lucchesi	Initial coding.

CONTENTS:

```
integer err
```

```

      if ( fmode .EQ. 0) then
          fid = ncopn (fname, NCWRITE, rc)
      else
          fid = ncopn (fname, NCNOWRIT, rc)
      endif
      if (err("Open: error opening file",rc,-39) .NE. 0) return

      rc = 0
      return
      end

!-----  

!           NASA/GSFC, Data Assimilation Office, Code 910.3, GEOS/DAS      !
!-----  


```

2.3 GFIO_PutVar – Write a variable to the file (Source File: *gfio.F*)

This routine is used to write a variable to an open GFIO stream. Multiple vertical levels can be written at one time provided they are contiguous in memory. Date and time must be consistent with the time increment and the starting date/time as defined in GFIO_Create. Times must fall on minute boundaries to allow GrADS to work. Error checking is done for dimensions that are out of bounds.

INTERFACE:

```

subroutine GFIO_PutVar ( fid, vname, yyyyymmdd, hhmmss,
&                           im, jm, kbeg, kount, grid,
&                           rc )

```

USES:

```

Implicit NONE
#include "netcdf.inc"
#include "gfio.h"

```

INPUT PARAMETERS:

integer	fid	! File handle
character(*)	vname	! Variable name
integer	yyyyymmdd	! Year-month-day, e.g., 19971003
integer	hhmmss	! Hour-minute-second, e.g., 120000
integer	im	! size of longitudinal dimension
integer	jm	! size of latitudinal dimension
integer	kbeg	! first level to write; if 2-D grid ! use kbeg = 0.
integer	kount	! number of levels to write

```
real           grid(im,jm,kount) ! Gridded data to write at this time
```

OUTPUT PARAMETERS:

```
integer          rc ! Error return code:
!   rc =  0  all is well
!   rc = -2  time is inconsistent with increment
!   rc = -3  number of levels is incompatible with file
!   rc = -4  im is incompatible with file
!   rc = -5  jm is incompatible with file
!   rc = -6  time must fall on a minute boundary
!   rc = -7  error in diffdate
!   rc = -12  error determining default precision
!   rc = -13  error determining variable type
!   rc = -15  data outside of valid range
!   rc = -16  data outside of packing range
!   rc = -17  data outside of pack and valid range
! NetCDF Errors
!
! -----
!   rc = -38  error from ncvpt (dimension variable)
!   rc = -40  error from ncvid
!   rc = -41  error from ncdid or ncinq (lat or lon)
!   rc = -42  error from ncdid or ncinq (lev)
!   rc = -43  error from ncvid (time variable)
!   rc = -44  error from ncagt (time attribute)
!   rc = -45  error from ncvpt
!   rc = -52  error from ncvinq
!   rc = -53  error from ncagtc/ncagt
```

REVISION HISTORY:

1997.10.13	da Silva/Lucchesi	Initial interface design.
1998.02.10	Lucchesi	Added support for applications running with 64-bit reals.
1998.03.30	Lucchesi	Documentation expanded. Clean-up of code.
1998.07.02	Lucchesi	Replaced vid with vname in argument list & made related mods to code.
1998.09.24	Lucchesi	Changed error codes, removed DIM_CHECK if-def
1998.10.27	Lucchesi	Added support for packing and range checks

CONTENTS:

```
integer timeid, dimSize, dimId
character*MAXCHR dimName
integer corner(4), edges(4)
```

```

integer vid
integer seconds, DiffDate, timeIndex
integer minutes                      ! added as a work-around
integer idx, i, j, k
integer begDate, begTime, timInc
integer err

! Variables for dealing with precision

real*4, allocatable :: grid_32(:,:,:)
real*8, allocatable :: grid_64(:,:,:)
real*4 dummy32
real*8 dummy64
real    dummy

! Variables for NCVINQ

character*MAXCHR varName
integer type, nvDims, vdims(MAXVDIMS), nvAtts

! Variables for packing and range checking

integer*2, allocatable :: grid_16(:,:,:)
real*4 high_32, low_32, amiss_32
real*4 scale_32, offset_32
logical :: outRange = .FALSE.
logical :: outPRange = .FALSE.

! Make NetCDF errors non-fatal, but issue warning messages.

call ncopt(NCVERBOS)

! Check to make sure max string lengths are large enough.  NetCDF defines
! MAXNCNAM, but it can't be used in a character*MAXNCNAM statement.
! MAXCHR is a CPP define in the gfio.h file.

if (MAXCHR .LT. MAXNCNAM) then
  print *, 'GFIO_PutVar warning: MAXNCNAM is larger than ',
          'dimName array size.'
endif

! Determine NetCDF variable ID.

vid = ncvid (fid, vname, rc)
if (err("PutVar: variable not defined",rc,-40) .NE. 0) return

! Basic error checking

```

```

dimId = ncdid (fid, 'lon', rc)
if (err("PutVar: can't get ID for lon",rc,-41) .NE. 0) return
call ncinq (fid, dimId, dimName, dimSize, rc)
if (err("PutVar: can't get info for lon",rc,-41) .NE. 0) return
if (dimSize .ne. im) then
  rc = -4
  return
endif

dimId = ncdid (fid, 'lat', rc)
if (err("PutVar: can't get ID for lat",rc,-41) .NE. 0) return
call ncinq (fid, dimId, dimName, dimSize, rc)
if (err("PutVar: can't get info for lat",rc,-41) .NE. 0) return
if (dimSize .ne. jm) then
  rc = -5
  return
endif

if (kbeg .NE. 0) then
  dimId = ncdid (fid, 'lev', rc)
  if (err("PutVar: can't get ID for lev",rc,-42) .NE. 0) return
  call ncinq (fid, dimId, dimName, dimSize, rc)
  if (err("PutVar: can't get info for lev",rc,-42) .NE. 0) return
  if (kbeg-1 + kount .gt. dimSize) then
    rc = -3
    return
  endif
endif

! Determine number of seconds since starting date/time.

timeId = ncvid (fid, 'time', rc)
if (err("PutVar: time not defined",rc,-43) .NE. 0) return
call ncagt (fid, timeId, 'begin_date', begDate, rc)
if (err("PutVar: missing begin_date",rc,-44) .NE. 0) return
call ncagt (fid, timeId, 'begin_time', begTime, rc)
if (err("PutVar: missing begin_time",rc,-44) .NE. 0) return

seconds = DiffDate (begDate, begTime, yyyyymmdd, hhmmss)

if (seconds .lt. 0) then
  print *, 'GFIO_PutVar: Error code from diffdate. Problem with',
  .      ' date/time.'
  rc = -7
  return
endif
if ( MOD (seconds,60) .eq. 0 ) then
  minutes = seconds / 60

```

```

    else
        print *, 'GFTIO_PutVar: Currently, times must fall on minute ',
        .      'boundaries.'
        rc = -6
        return
    endif

! Confirm that this time is consistent with the starting time coupled with
! the time increment.

    call ncagt (fid, timeId, 'time_increment', timInc, rc)
    if (err("PutVar: missing time increment",rc,-44) .NE. 0) return

    if ( MOD (seconds, timInc) .ne. 0 ) then
        print *, 'GFTIO_putvar: Absolute time of ',seconds,' not ',
        .      'possible with an interval of ',timInc
        rc = -2
        return
    else
        timeIndex = seconds/timInc + 1
    endif

! Load starting indicies.

    if ( kbeg .eq. 0 ) then
        corner(1)=1
        corner(2)=1
        corner(3)=timeIndex
        edges(1)=im
        edges(2)=jm
        edges(3)=1
    else
        corner(1)=1
        corner(2)=1
        corner(3)=kbeg
        corner(4)=timeIndex
        edges(1)=im
        edges(2)=jm
        edges(3)=kount
        edges(4)=1
    endif

! Check variable against valid range.

    call ncagt (fid, vid, 'vmin', low_32, rc)
    if (err("PutVar: can't get vmin",rc,-53) .NE. 0) return
    call ncagt (fid, vid, 'vmax', high_32, rc)
    if (err("PutVar: can't get vmax",rc,-53) .NE. 0) return

```

```

call ncagt (fid, vid, 'fmissing_value', amiss_32, rc)
if (err("PutVar: can't get fmissing_value",rc,-53) .NE. 0) return
if (low_32 .NE. amiss_32 .OR. high_32 .NE. amiss_32) then
  do k=1,kount
    do j=1,jm
      do i=1,im
        if (grid(i,j,k) .GT. high_32 .OR. grid(i,j,k) .LT.
. low_32) then
          outRange = .TRUE.
          goto 100
        endif
      enddo
    enddo
  enddo
100   continue
endif

! Determine if we are writing single- or double-precision.

call ncvinq (fid, vid, varName, type, nvDims, vDims, nvAtts, rc)
if (err("PutVar: error in variable inquire",rc,-52) .NE. 0) return

! Write variable in the appropriate precision.

if (HUGE(dummy) .EQ. HUGE(dummy32)) then           ! -r4
  if (type .EQ. NCFLOAT) then                      ! 32-bit
    call ncvpt (fid, vid, corner, edges, grid, rc)
  else if (type .EQ. NCDOUBLE) then                ! 64-bit
    allocate (grid_64(im,jm,kount))
    do k=1,kount
      do j=1,jm
        do i=1,im
          grid_64(i,j,k) = grid(i,j,k)
        enddo
      enddo
    enddo
    call ncvpt (fid, vid, corner, edges, grid_64, rc)
    deallocate (grid_64)
  else if (type .EQ. NCSHORT) then
    call ncagt (fid, vid, 'packmax', high_32, rc)
    if (err("PutVar: error getting packmax",rc,-53) .NE. 0) return
    call ncagt (fid, vid, 'packmin', low_32, rc)
    if (err("PutVar: error getting packmin",rc,-53) .NE. 0) return
    call ncagt (fid, vid, 'scale_factor', scale_32, rc)
    if (err("PutVar: error getting scale",rc,-53) .NE. 0) return
    call ncagt (fid, vid, 'add_offset', offset_32, rc)
    if (err("PutVar: error getting offset",rc,-53) .NE. 0) return
    allocate (grid_16(im,jm,kount))
  endif
endif

```

```

      do k=1,kount
        do j=1,jm
          do i=1,im
            if ( grid(i,j,k) .LT. low_32 .OR. grid(i,j,k) .GT.
.           high_32) then
              grid_16(i,j,k) = PACK_FILL
              outPRange = .TRUE.
            else
              grid_16(i,j,k) = (grid(i,j,k) - offset_32)/scale_32
            endif
          enddo
        enddo
      call ncvt (fid, vid, corner, edges, grid_16, rc)
      deallocate (grid_16)
    else
      rc = -13
      return
    endif
  else if (HUGE(dummy) .EQ. HUGE(dummy64)) then ! -r8
    if (type .EQ. NCFLOAT) then                      ! 32-bit
      allocate (grid_32(im,jm,kount))
      do k=1,kount
        do j=1,jm
          do i=1,im
            grid_32(i,j,k) = grid(i,j,k)
          enddo
        enddo
      call ncvt (fid, vid, corner, edges, grid_32, rc)
      deallocate (grid_32)
    else if (type .EQ. NCDOUBLE) then                ! 64-bit
      call ncvt (fid, vid, corner, edges, grid, rc)
    else if (type .EQ. NCSHORT) then
      call ncagt (fid, vid, 'packmax', high_32, rc)
      if (err("PutVar: error getting packmax",rc,-53) .NE. 0) return
      call ncagt (fid, vid, 'packmin', low_32, rc)
      if (err("PutVar: error getting packmin",rc,-53) .NE. 0) return
      call ncagt (fid, vid, 'scale_factor', scale_32, rc)
      if (err("PutVar: error getting scale",rc,-53) .NE. 0) return
      call ncagt (fid, vid, 'add_offset', offset_32, rc)
      if (err("PutVar: error getting offset",rc,-53) .NE. 0) return
      allocate (grid_16(im,jm,kount))
      do k=1,kount
        do j=1,jm
          do i=1,im
            if ( grid(i,j,k) .LT. low_32 .OR. grid(i,j,k) .GT.
.           high_32) then

```

```

        grid_16(i,j,k) = PACK_FILL
    else
        grid_16(i,j,k) = (grid(i,j,k) - offset_32)/scale_32
    endif
    enddo
    enddo
    enddo
    call ncvpt (fid, vid, corner, edges, grid_16, rc)
    deallocate (grid_16)
else
    rc = -13
    return
endif
else
    rc = -12
    return
endif
if (err("PutVar: error writing variable",rc,-45) .NE. 0) return

! Write time to file.

corner(1)=timeIndex
edges(1)=1
call ncvpt (fid,timeId,corner,edges,minutes,rc)
if (err("PutVar: error writing time",rc,-38) .NE. 0) return

if (outRange .AND. outRange) then
    rc = -17
else if (outPRange) then
    rc = -16
else if (outRange) then
    rc = -15
else
    rc = 0
endif

return
end

!-----!
!      NASA/GSFC, Data Assimilation Office, Code 910.3, GEOS/DAS      !
!-----!
```

2.4 GFIO_GetVar – Read a variable from the file (Source File: *gfio.F*)

This routine will read one or more levels of "vname" into the buffer passed in as "grid." "fid" is the file handle returned by Gfio_open.

INTERFACE:

```
subroutine GFIO_GetVar ( fid, vname, yyyyymmdd, hhmmss,
& im, jm, kbeg, kount, grid, rc )
```

USES:

```
Implicit NONE
#include "netcdf.inc"
#include "gfio.h"
```

INPUT PARAMETERS:

integer	fid	! File handle
character(*)	vname	! Variable name
integer	yyyyymmdd	! Year-month-day, e.g., 19971003
integer	hhmmss	! Hour-minute-second, e.g., 120000
integer	im	! size of longitudinal dimension
integer	jm	! size of latitudinal dimension
integer	kbeg	! first level to read; if 2-D grid ! set kbeg = 0.
integer	kount	! number of levels to read

OUTPUT PARAMETERS:

```
real grid(im,jm,kount) ! Gridded data read for this time
integer rc      ! Error return code:
                  ! rc = 0 all is well
                  ! rc = -2 time is inconsistent with increment
                  ! rc = -3 number of levels is incompatible with file
                  ! rc = -4 im is incompatible with file
                  ! rc = -5 jm is incompatible with file
                  ! rc = -6 time must fall on a minute boundary
                  ! rc = -7 error in diffdate
                  ! rc = -12 error determining default precision
                  ! rc = -13 error determining variable type
                  ! NetCDF Errors
                  ! -----
                  ! rc = -38 error from ncvpt (dimension variable)
                  ! rc = -40 error from ncvid
                  ! rc = -41 error from ncwid or ncwidq (lat or lon)
                  ! rc = -42 error from ncwid or ncwidq (lev)
                  ! rc = -43 error from ncvid (time variable)
                  ! rc = -44 error from ncagt (time attribute)
                  ! rc = -46 error from ncvgt
```

REVISION HISTORY:

1997.10.13 da Silva/Lucchese	Initial interface design.
1998.07.07 Lucchesi	Combined two GetVar routines into this one.
1998.09.24 Lucchesi	Updated error codes.

CONTENTS:

```

integer timeId, begDate, begTime, seconds, minutes, timInc
integer corner(4), edges(4), timeIndex
integer vid
integer DiffDate
integer dimSize, dimId
character*MAXCHR dimName
integer err
integer i,j,k

```

! Variables for dealing with precision

```

real*4, allocatable :: grid_32(:,:,:)
real*8, allocatable :: grid_64(:,:,:)
real*4 dummy32
real*8 dummy64
real    dummy

```

! Variables for NCVINQ

```

character*MAXCHR varName
integer type, nvDims, vdims(MAXVDIMS), nvAtts

```

! Variables for packing

```

integer*2, allocatable :: grid_16(:,:,:)
integer*2 amiss_16
real*4 amiss_32
real*4 scale_32, offset_32

```

! Make NetCDF errors non-fatal, but issue warning messages.

```
call ncpopt(NCVERBOS)
```

! Check to make sure max string lengths are large enough. NetCDF defines
! MAXNCNAM, but it can't be used in a character*MAXNCNAM statement.
! MAXCHR is a CPP define in the gfio.h file.

```

if (MAXCHR .LT. MAXNCNAM) then
  print *, 'GFIO_GetVar warning: MAXNCNAM is larger than ',

```

```

        'dimName array size.'
      endif

! Determine NetCDF variable ID.

  vid = ncvid (fid, vname, rc)
  if (err("GetVar: variable not defined",rc,-40) .NE. 0) return

! Optional dimension checks.

  dimId = ncdid (fid, 'lon', rc)
  if (err("GetVar: can't get ID for lon",rc,-41) .NE. 0) return
  call ncdinq (fid, dimId, dimName, dimSize, rc)
  if (err("GetVar: can't get info for lon",rc,-41) .NE. 0) return
  if (dimSize .ne. im) then
    rc = -4
    return
  endif

  dimId = ncdid (fid, 'lat', rc)
  if (err("GetVar: can't get ID for lat",rc,-41) .NE. 0) return
  call ncdinq (fid, dimId, dimName, dimSize, rc)
  if (err("GetVar: can't get info for lat",rc,-41) .NE. 0) return
  if (dimSize .ne. jm) then
    rc = -5
    return
  endif

  if (kbeg .NE. 0) then
    dimId = ncdid (fid, 'lev', rc)
    if (err("GetVar: can't get ID for lev",rc,-42) .NE. 0) return
    call ncdinq (fid, dimId, dimName, dimSize, rc)
    if (err("GetVar: can't get info for lev",rc,-42) .NE. 0) return
    if (kbeg-1 + kount .gt. dimSize) then
      rc = -3
      return
    endif
  endif

! Get beginning time & date. Calculate offset seconds from start.

  timeId = ncvid (fid, 'time', rc)
  if (err("GetVar: time not defined",rc,-43) .NE. 0) return
  call ncagt (fid, timeId, 'begin_date', begDate, rc)
  if (err("GetVar: missing begin_date",rc,-44) .NE. 0) return
  call ncagt (fid, timeId, 'begin_time', begTime, rc)
  if (err("GetVar: missing begin_time",rc,-44) .NE. 0) return

```

```

seconds = DiffDate (begDate, begTime, yyyyymmdd, hhmmss)

! Make sure input time are valid.

if (seconds .lt. 0) then
  print *, 'GFIO_GetVar: Error code from diffdate. Problem with',
  ' date/time.'
  rc = -7
  return
endif
if ( MOD (seconds,60) .eq. 0 ) then
  minutes = seconds / 60
else
  print *, 'GFIO_GetVar: Currently, times must fall on minute ',
  'boundaries.'
  rc = -6
  return
endif

! Determine the time index from the offset and time increment.

call ncagt (fid, timeId, 'time_increment', timInc, rc)
if (err("GetVar: missing time increment",rc,-44) .NE. 0) return

if ( MOD (seconds, timInc) .ne. 0 ) then
  print *, 'GFIO_getvar: Absolute time of ',seconds,' not ',
  'possible with an interval of ',timInc
  rc = -2
  return
else
  timeIndex = seconds/timInc + 1
endif

! Load starting indicies.

if ( kbeg .eq. 0 ) then
  corner(1)=1
  corner(2)=1
  corner(3)=timeIndex
  edges(1)=im
  edges(2)=jm
  edges(3)=1
else
  corner(1)=1
  corner(2)=1
  corner(3)=kbeg
  corner(4)=timeIndex
  edges(1)=im

```

```

edges(2)=jm
edges(3)=kount
edges(4)=1
endif

! Determine data type.

call ncvinq (fid, vid, varName, type, nvDims, vDims, nvAtts, rc)
if (err("GetVar: error in variable inquire",rc,-52) .NE. 0) return

! Read variable in the appropriate precision.

if (HUGE(dummy) .EQ. HUGE(dummy32)) then           ! -r4
  if (type .EQ. NCFLOAT) then                      ! 32-bit
    call ncvgt (fid, vid, corner, edges, grid, rc)
  else if (type .EQ. NCDOUBLE) then                ! 64-bit
    print *, 'DEBUG: getvar NCDOUBLE'
    allocate (grid_64(im,jm,kount))
    call ncvgt (fid, vid, corner, edges, grid_64, rc)
    do k=1,kount
      do j=1,jm
        do i=1,im
          grid(i,j,k) = grid_64(i,j,k)
        enddo
      enddo
    enddo
    deallocate (grid_64)
  else if (type .EQ. NCSHORT) then
    call ncagt (fid, vid, 'scale_factor', scale_32, rc)
    if (err("PutVar: error getting scale",rc,-53) .NE. 0) return
    call ncagt (fid, vid, 'add_offset', offset_32, rc)
    if (err("PutVar: error getting offset",rc,-53) .NE. 0) return
    call ncagt (fid, vid, 'missing_value', amiss_16, rc)
    if (err("PutVar: error getting offset",rc,-53) .NE. 0) return
    call ncagt (fid, vid, 'fmissing_value', amiss_32, rc)
    if (err("PutVar: error getting offset",rc,-53) .NE. 0) return
    allocate (grid_16(im,jm,kount))
    call ncvgt (fid, vid, corner, edges, grid_16, rc)
    do k=1,kount
      do j=1,jm
        do i=1,im
          if ( grid_16(i,j,k) .EQ. amiss_16 ) then
            grid(i,j,k) = amiss_32
          else
            grid(i,j,k) = scale_32*grid_16(i,j,k) + offset_32
          endif
        enddo
      enddo
    enddo
  endif
endif

```

```

        enddo
        deallocate (grid_16)
    else
        rc = -13
        return
    endif
else if (HUGE(dummy) .EQ. HUGE(dummy64)) then ! -r8
    if (type .EQ. NCFLOAT) then                  ! 32-bit
        allocate (grid_32(im,jm,kount))
        call ncvgt (fid, vid, corner, edges, grid_32, rc)
        do k=1,kount
            do j=1,jm
                do i=1,im
                    grid(i,j,k) = grid_32(i,j,k)
                enddo
            enddo
        enddo
        deallocate (grid_32)
    elseif (type .EQ. NCDOUBLE) then           ! 64-bit
        call ncvgt (fid, vid, corner, edges, grid, rc)
    else if (type .EQ. NCSHORT) then
        call ncagt (fid, vid, 'scale_factor', scale_32, rc)
        if (err("PutVar: error getting scale",rc,-53) .NE. 0) return
        call ncagt (fid, vid, 'add_offset', offset_32, rc)
        if (err("PutVar: error getting offset",rc,-53) .NE. 0) return
        call ncagt (fid, vid, 'missing_value', amiss_16, rc)
        if (err("PutVar: error getting offset",rc,-53) .NE. 0) return
        call ncagt (fid, vid, 'fmissing_value', amiss_32, rc)
        if (err("PutVar: error getting offset",rc,-53) .NE. 0) return
        allocate (grid_16(im,jm,kount))
        call ncvgt (fid, vid, corner, edges, grid_16, rc)
        do k=1,kount
            do j=1,jm
                do i=1,im
                    if ( grid_16(i,j,k) .EQ. amiss_16 ) then
                        grid(i,j,k) = amiss_32
                    else
                        grid(i,j,k) = scale_32*grid_16(i,j,k) + offset_32
                    endif
                enddo
            enddo
        enddo
        deallocate (grid_16)
    else
        rc = -13
        return
    endif
else

```

```

rc = -12
return
endif
if (err("GetVar: error reading variable",rc,-46) .NE. 0) return

rc = 0
return
end

!-----  

!      NASA/GSFC, Data Assimilation Office, Code 910.3, GEOS/DAS      !
!-----
```

2.5 GFIO_DimInquire – Gets dimension information from a GFIO file. (Source File: *gfio.F*)

This routine is used to get dimension information from an existing GFIO file. This dimension information can subsequently be used to allocate arrays for reading data from the file. For more complete information about the contents of a file, *Gfio_Inquire* should be used.

INTERFACE:

```
subroutine GFIO_DimInquire (fid,im,jm,km,lm,nvars,ngatts,rc)
```

USES:

```

Implicit NONE
#include "netcdf.inc"
#include "gfio.h"
```

INPUT PARAMETERS:

```
integer          fid           ! File handle
```

OUTPUT PARAMETERS:

```

integer          im            ! Size of longitudinal dimension
integer          jm            ! Size of latitudinal dimension
integer          km            ! Size of vertical dimension
                           !   km=0 if surface-only file
integer          lm            ! Number of times
integer          nvars          ! Number of variables
integer          ngatts         ! Number of global attributes
integer          rc            ! Error return code:
                           !   rc = 0    all is well
                           !   NetCDF Errors
                           !   -----
```

```

      !  rc = -41  error from ncdid or ncinq (lat or lon)
      !  rc = -42  error from ncdid or ncinq (lev)
      !  rc = -43  error from ncvid (time variable)
      !  rc = -47  error from ncdid or ncinq (time)
      !  rc = -48  error from ncinq

```

REVISION HISTORY:

1998.07.02	Lucchesi	Initial interface design.
1998.08.05	Lucchesi	Added "ngatts"
1998.09.24	Lucchesi	Revamped error codes

CONTENTS:

```

integer timeid, dimId
character*MAXCHR dimName
integer nDims
integer err
logical :: surfaceOnly = .FALSE.

! Make NetCDF errors non-fatal, but issue warning messages.

call ncpopt(NCVERBOS)

! Check FID here.

! Check to make sure max string lengths are large enough.  NetCDF defines
! MAXNCNAM, but it can't be used in a character*MAXNCNAM statement.
! MAXCHR is a CPP define in the gfio.h file.

if (MAXCHR .LT. MAXNCNAM) then
  print *, 'GFIO_DimInquire warning: MAXNCNAM is larger than ',
  .          'dimName array size.'
endif

! Get basic information from file.

call ncinq (fid, nDims, nvars, ngatts, dimId, rc)
if (err("DimInquire: ncinq failed",rc,-48) .NE. 0) return
nvars = nvars - nDims

if (nDims .EQ. 3) then
  surfaceOnly = .TRUE.

```

```

        print *, 'THIS IS A SURFACE FILE.'
        endif

! Extract dimension information

dimId = ncdid (fid, 'lon', rc)
if (err("DimInquire: can't get ID for lon",rc,-41) .NE. 0) return
call ncdinq (fid, dimId, dimName, im, rc)
if (err("DimInquire: can't get info for lon",rc,-41) .NE. 0) return

dimId = ncdid (fid, 'lat', rc)
if (err("DimInquire: can't get ID for lat",rc,-41) .NE. 0) return
call ncdinq (fid, dimId, dimName, jm, rc)
if (err("DimInquire: can't get info for lat",rc,-41) .NE. 0) return

if (.NOT. surfaceOnly) then
    dimId = ncdid (fid, 'lev', rc)
    if (err("DimInquire: can't get ID for lev",rc,-42) .NE. 0) return
    call ncdinq (fid, dimId, dimName, km, rc)
    if (err("DimInquire: can't get info for lev",rc,-42) .NE. 0)
        .
        return
else
    km=0
endif

dimId = ncdid (fid, 'time', rc)
if (err("DimInquire: can't get ID for time",rc,-47) .NE. 0) return
call ncdinq (fid, dimId, dimName, lm, rc)
if (err("DimInquire: can't get info for time",rc,-47) .NE. 0) return

rc=0
return
end

!-----  

!      NASA/GSFC, Data Assimilation Office, Code 910.3, GEOS/DAS      !
!-----
```

2.6 GFIO_Inquire – Get information about a GFIO file. (Source File: *gfio.F*)

This routine is used to get as much information as possible about the contents of a GFIO file. The file handle (fid) is passed in and detailed information about dimensions and variables are returned to the application. A simpler inquire routine for dimension information is Gfio_DimInquire.

INTERFACE:

```
subroutine GFIO_Inquire ( fid, im, jm, km, lm, nvars,
&                           title, source, contact, amiss,
&                           lon, lat, levs, levunits,
&                           yyyyymmdd, hhmmss, timinc,
&                           vname, vtitle, vunits, kmvar,
&                           valid_range , packing_range, rc)
```

USES:

```
Implicit NONE
#include "netcdf.inc"
#include "gfio.h"
```

INPUT PARAMETERS:

integer	fid	! ----- Global Metadata ----- ! File handle from GFIO_open
---------	-----	---

INPUT/OUTPUT PARAMETERS:

integer	im	! size of longitudinal dimension
integer	jm	! size of latitudinal dimension
integer	km	! size of vertical dimension
integer	lm	! size of time dimension
		! On input, (im,jm,km,lm) contains the
		! size of arrays (lon,lat,lev,yyyyymmdd)
		! as declared in the calling program.
		! On output, (im,jm,km,lm) contains the
		! size of the coordinate variables
		! (lon,lat,lev,yyyyymmdd) on file.
integer	nvars	! number of variables on file

OUTPUT PARAMETERS:

character(*)	title	! Title of the data set
character(*)	source	! Where it came from
character(*)	contact	! Who to contact about the data set
real	amiss	! Missing value
! ----- Dimension Metadata -----		
real	lon(im)	! longitude of center of gridbox in
		! degrees east of Greenwich (can be
		! -180 -> 180 or 0 -> 360)
real	lat(jm)	! latitude of center of gridbox in
		! degrees north of equator
real	levs(km)	! Level (units given by levunits) of
		! center of gridbox

```

character(*)  levunits      ! units of level dimension, e.g.,
                           !   "hPa", "sigma_level"
integer       yyymmdd(lm)   ! Year-month-day on file
integer       hhmmss(lm)   ! Hour-minute-second on file
integer       timinc       ! Time increment.

                           ! ----- Variable Metadata -----
integer       nvars        ! number of variables on file
character(*) vname(nvars)  ! variable short name, e.g., "hght"
character(*) vtitle(nvars) ! variable long name, e.g.,
                           !   "Geopotential Height"
character(*) vunits(nvars) ! variable units, e.g., "meter/second"
integer       kmvar(nvars)  ! number of levels for variable; it can
                           ! either be 0 (2-D fields) or equal to km
real        valid_range(2,nvars) ! Variable valid range; set to
                           !   "amiss" if not known.

                           ! ----- Packing Metadata -----
real    packing_range(2,nvars) ! Variable packing range used
                           ! for 16-bit packing. If packing was not
                           ! used then returned values will be amiss.
                           ! NOTE: all unpacking is done transparently
                           !       by GFIO_GetVar().

integer       rc           ! Error return code:
                           !   rc = 0      all is well
                           !   rc = -3     number of levels is incompatible with file
                           !   rc = -4     im is incompatible with file
                           !   rc = -5     jm is incompatible with file
                           !   rc = -8     lm is incompatible with file
                           !   rc = -9     nvars is incompatible with file
                           !   rc = -9     vname strings too short
                           !   rc = -14    error in getdate
                           ! NetCDF Errors
                           !
                           ! -----
                           !   rc = -41   error from ncwid or ncinq (lat or lon)
                           !   rc = -42   error from ncwid or ncinq (lev)
                           !   rc = -43   error from ncvid (time variable)
                           !   rc = -44   error from ncagt (time attribute)
                           !   rc = -47   error from ncwid or ncinq (time)
                           !   rc = -48   error from ncinq
                           !   rc = -50   error from ncagt (level attribute)
                           !   rc = -51   error from ncagt/ncagt (global attribute)
                           !   rc = -52   error from ncvinq
                           !   rc = -53   error from ncagt/ncagt

```

!FUTURE ENHANCEMENT:

Next release should include a flag for precision.

REVISION HISTORY:

1998.07.02	Lucchesi	Initial interface design.
1998.07.17	Lucchesi	Initial coding.
1998.10.09	Lucchesi	Restructured return codes.

CONTENTS:

!-----

! Local Variables

```

integer vid(nvars)
integer timeId, latId, lonId, levId
integer nDims, recdim, ngatts
integer varType, nvDims, vDims(MAXVDIMS), nvAtts
integer yyyyymmdd_beg, hhmmss_beg, hour, min, sec
integer start1D, minutes(lm)
character*8 strBuf
character*MAXCHR dimName
integer dimSize
integer i
logical :: surfaceOnly = .FALSE.

! REAL*4 variables for 32 bit input from netCDF file.

real*4 lon_32(im), lat_32(jm), levs_32(km)
real*4 amiss_32
real*4 pRange_32(2,nvars),vRange_32(2,nvars)
integer err

! Make NetCDF errors non-fatal, but issue warning messages.

call ncpopt(NCVERBOS)

! Check length of vname string

if (LEN(vname(1)) .lt. MAXNCNAM) then
  print *, 'GFIO_Inquire: length of vname array must be at least ',
  .          MAXNCNAM, ' bytes.'
  rc = -9
  return
endif

! Check to make sure max string lengths are large enough. NetCDF defines
! MAXNCNAM, but it can't be used in a character*MAXNCNAM statement.
! MAXCHR is a CPP define in the gfio.h file.

if (MAXCHR .LT. MAXNCNAM) then

```

```

        print *, 'GFIO_GetVar warning: MAXNCNAM is larger than ',
        .      'dimName array size.'
      endif

! Get basic information from the file

call ncinq (fid,nDims,dimSize,ngatts,reclim,rc)
if (err("Inquire: ncinq failed",rc,-48) .NE. 0) return

dimSize = dimSize - nDims
if (dimSize .NE. nvars) then
  rc = -9
  nvars = dimSize
  return
endif

if ( nDims .EQ. 3) then
  surfaceOnly = .TRUE.
endif

! Dimension error checking.

lonId = ncdid (fid, 'lon', rc)
if (err("Inquire: can't get ID for lon",rc,-41) .NE. 0) return
call ncdinq (fid, lonId, dimName, dimSize, rc)
if (err("Inquire: can't get info for lon",rc,-41) .NE. 0) return
if (dimSize .ne. im) then
  rc = -4
  im = dimSize
  return
endif

latId = ncdid (fid, 'lat', rc)
if (err("Inquire: can't get ID for lat",rc,-41) .NE. 0) return
call ncdinq (fid, latId, dimName, dimSize, rc)
if (err("Inquire: can't get info for lat",rc,-41) .NE. 0) return
if (dimSize .ne. jm) then
  rc = -5
  jm = dimSize
  return
endif

if (.NOT. surfaceOnly) then
  levId = ncdid (fid, 'lev', rc)
  if (err("Inquire: can't get ID for lev",rc,-42) .NE. 0) return
  call ncdinq (fid, levId, dimName, dimSize, rc)
  if (err("Inquire: can't get info for lev",rc,-42) .NE. 0) return
  if (km .ne. dimSize) then

```

```

    rc = -3
    km = dimSize
    return
  endif
endif

timeId = ncdid (fid, 'time', rc)
if (err("Inquire: can't get ID for time",rc,-47) .NE. 0) return
call ncdinq (fid, timeId, dimName, dimSize, rc)
if (err("Inquire: can't get info for time",rc,-47) .NE. 0) return
if (lm .ne. dimSize) then
  rc = -8
  lm = dimSize
  return
endif

call ncinq (fid,nDims,dimSize,ngatts,reclim,rc)
if (err("Inquire: ncinq failed",rc,-48) .NE. 0) return
dimSize = dimSize - nDims
if (dimSize .NE. nvars) then
  rc = -9
  nvars = dimSize
  return
endif

start1D=1

! Get dimension values (coordinates)

! Dimension values in netCDF file are 32 bit.

call ncvgt (fid,lonId,start1D,im,lon_32,rc)
if (err("Inquire: error reading lons",rc,-49) .LT. 0) return
do i=1,im
  lon(i)=lon_32(i)
enddo

call ncvgt (fid,latId,start1D,jm,lat_32,rc)
if (err("Inquire: error reading lats",rc,-49) .LT. 0) return
do i=1,jm
  lat(i)=lat_32(i)
enddo

if (.NOT. surfaceOnly) then
  call ncvgt (fid,levId,start1D,km,levs_32,rc)
  if (err("Inquire: error reading levs",rc,-49) .LT. 0) return
  do i=1,km
    levs(i)=levs_32(i)
  enddo
end

```

```

        enddo
      endif

      call ncvgt (fid,timeId,start1D,lm,minutes,rc)
      if (err("Inquire: error reading times",rc,-49) .LT. 0) return

! Get dimension attributes.

      if (.NOT. surfaceOnly) then
        call ncagtc (fid,levid,'units',levunits,LEN(levunits),rc)
        if (err("Inquire: error reading lev units",rc,-50) .LT. 0)
          .
          return
      endif

      call ncagt (fid,timeid,'time_increment',timinc,rc)
      if (err("Inquire: missing time increment",rc,-44) .LT. 0) return

      call ncagt (fid,timeid,'begin_date',yyyymmdd_beg,rc)
      if (err("Inquire: missing begin_date",rc,-44) .NE. 0) return

      call ncagt (fid,timeid,'begin_time',hhmmss_beg,rc)
      if (err("Inquire: missing begin_time",rc,-44) .NE. 0) return

! Calculate and load YYYYMMDD and HHMMSS values.

      do i=1,lm
        call GetDate (yyyymmdd_beg,hhmmss_beg,minutes(i)*60,
                      yyyymmdd(i),hhmmss(i),rc)
        if (rc .LT. 0) then
          print *, "GFTIO_Inquire: error in getdate"
          rc = -14
          return
        endif
      enddo

! Get global attributes

      call ncagtc (fid,NCGLOBAL,'Title',title,LEN(title),rc)
      if (err("Inquire: missing Title",rc,-51) .NE. 0) return
      call ncagtc (fid,NCGLOBAL,'Source',source,LEN(source),rc)
      if (err("Inquire: missing Source",rc,-51) .NE. 0) return
      call ncagtc (fid,NCGLOBAL,'Contact',contact,LEN(contact),rc)
      if (err("Inquire: missing Contact",rc,-51) .NE. 0) return

! Get missing value. GFTIO forces this to be the same for all variables.

      call ncagt (fid, nDims+1,'fmissing_value',amiss_32,rc)
      if (err("Inquire: error getting missing value",rc,-53) .NE. 0)

```

```

.      return
amiss = amiss_32

! Get variable information.

do i=1,nvars
  vid(i)=nDims+i
  call ncvinq (fid,vid(i),vname(i),varType,nvDims,vDims,
.           nvAtts,rc)
  if (err("Inquire: variable inquire error",rc,-52) .NE. 0) return
  if (nvDims .EQ. 3) then
    kmvar(i)=0
  else
    kmvar(i)=km
  endif
  call ncagtc (fid, vid(i), 'long_name', vtitle(i),
.           LEN(vtitle(i)), rc)
  if (err("Inquire: variable attribute error",rc,-53) .NE. 0)
.   return
  call ncagtc (fid, vid(i), 'units', vunits(i), LEN(vunits(i)),
.           rc)
  if (err("Inquire: variable attribute error",rc,-53) .NE. 0)
.   return

! Get packing ranges and valid ranges. Errors are not fatal
! since these attributes are optional.

call ncpopt(0)
call ncagt (fid, vid(i), 'packmin', pRange_32(1,i), rc)
if (rc .NE. 0) then
  packing_range(1,i) = amiss
else
  packing_range(1,i) = pRange_32(1,i)
endif
call ncagt (fid, vid(i), 'packmax', pRange_32(2,i), rc)
if (rc .NE. 0) then
  packing_range(2,i) = amiss
else
  packing_range(2,i) = pRange_32(2,i)
endif
call ncagt (fid, vid(i), 'vmin', vRange_32(1,i), rc)
if (rc .NE. 0) then
  valid_range(1,i) = amiss
else
  valid_range(1,i) = vRange_32(1,i)
endif
call ncagt (fid, vid(i), 'vmax', vRange_32(2,i), rc)
if (rc .NE. 0) then

```

```

        valid_range(2,i) = amiss
    else
        valid_range(2,i) = vRange_32(2,i)
    endif
    call ncpopt(NCVERBOS)

enddo

rc=0
return
end

!-----  

!      NASA/GSFC, Data Assimilation Office, Code 910.3, GEOS/DAS      !
!-----
```

2.7 GFIO_Close – Closes file (Source File: *gfio.F*)

This routine is used to close an open GFIO stream.

INTERFACE:

```
subroutine GFIO_Close ( fid, rc )
```

USES:

```
Implicit NONE
#include "netcdf.inc"
#include "gfio.h"
```

INPUT PARAMETERS:

```
integer          fid           ! File handle
```

OUTPUT PARAMETERS:

```
integer          rc            ! Error return code:
                                !   rc = 0    all is well
                                !   NetCDF Errors
                                ! -----
                                !   rc = -54  error from ncclos (file close)
```

REVISION HISTORY:

1997.10.13 da Silva/Lucchesi	Initial interface design.
1998.03.30 Lucchesi	Documentation expanded. Clean-up of code. Added rc.

CONTENTS:

```
!-----
integer i
integer err

! Make NetCDF errors non-fatal, but issue warning messages.

call ncpopt(NCVERBOS)

call ncclos (fid, rc)
if (err("Close: error closing file",rc,-54) .NE. 0) return

rc = 0
return
end

!-----  

! NASA/GSFC, Data Assimilation Office, Code 910.3, GEOS/DAS !  

!-----
```

2.8 GFIO_PutIntAtt – Write a user-defined integer attribute (Source File: *gfio.F*)

This routine allows the user to define an integer attribute in an open GFIO file.

INTERFACE:

```
subroutine GFIO_PutIntAtt ( fid, name, count, buf, prec, rc )
```

USES:

```
Implicit NONE
#include "netcdf.inc"
#include "gfio.h"
```

INPUT PARAMETERS:

```
integer      fid      ! File handle
character*(*) name    ! Name of attribute
integer      count    ! Number of integers to write
integer      buf(count) ! Buffer with integer values
integer      prec    ! Desired precision of attribute value:
                      !   0 = 32 bit
                      !   1 = 64 bit
```

OUTPUT PARAMETERS:

```

integer      rc      ! Error return code:
!   rc = 0    all is well
!   rc = -12   error determining default precision
!   NetCDF Errors
! -----
!   rc = -36   error from ncaptc/ncapt (global attribute)
!   rc = -55   error from ncredf (enter define mode)
!   rc = -56   error from ncdef (exit define mode)

```

REVISION HISTORY:

1998.07.30	Lucchesi	Initial interface design.
1998.07.30	Lucchesi	Initial coding.
1998.09.24	Lucchesi	Changed error handling.
1998.09.28	Lucchesi	Added support for multiple precisions

CONTENTS:

```

integer*4 dummy32
integer*8 dummy64
integer i

integer*4, allocatable :: buf32(:)
integer*8, allocatable :: buf64(:)
integer err

call ncredf ( fid, rc )
if (err("PutIntAtt: could not enter define mode",rc,-55) .NE. 0)
.   return

if ( HUGE(dummy32) .EQ. HUGE(i) .AND. prec .EQ. 0 ) then      ! -i4
  call ncapt ( fid, NCGLOBAL, name, NCLONG, count, buf, rc ) ! 32-bit out

else if ( HUGE(dummy32) .EQ. HUGE(i) .AND. prec .EQ. 1 ) then ! -i4
  allocate ( buf64(count) )                                     ! 64-bit out
  do i=1,count
    buf64(i) = buf(i)
  enddo
  call ncapt ( fid, NCGLOBAL, name, NCDOUBLE, count, buf64, rc )
  deallocate (buf64)

else if ( HUGE(dummy64) .EQ. HUGE(i) .AND. prec .EQ. 0 ) then ! -i8
  allocate ( buf32(count) )                                     ! 32-bit out
  do i=1,count
    buf32(i) = buf(i)
  enddo

```

```

call ncapt ( fid, NCGLOBAL, name, NCLONG, count, buf32, rc )
deallocate (buf32)

else if (HUGE(dummy64) .EQ. HUGE(i) .AND. prec .EQ. 1 ) then ! -i8
    call ncapt ( fid, NCGLOBAL, name, NCDOUBLE, count, buf, rc ) ! 64-bit out

else
    rc = -12
    return
endif
if (err("PutIntAtt: error writing attribute",rc,-36) .NE. 0)
.   return

call ncendf ( fid, rc )
if (err("PutIntAtt: could not exit define mode",rc,-56) .NE. 0)
.   return

rc = 0
return
end

!-----
!      NASA/GSFC, Data Assimilation Office, Code 910.3, GEOS/DAS      !
!-----
```

2.9 GFIO_PutRealAtt – Write a user-defined real attribute (Source File: *gfio.F*)

This routine allows the user to define a real attribute in an open GFIO file.

INTERFACE:

```
subroutine GFIO_PutRealAtt ( fid, name, count, buf, prec, rc )
```

USES:

```
Implicit NONE
#include "netcdf.inc"
#include "gfio.h"
```

INPUT PARAMETERS:

```

integer      fid          ! File handle
character(*) name        ! Name of attribute
integer      count        ! Number of integers to write
real         buf(count)  ! Buffer with real values
integer      prec        ! Desired precision of attribute value:
                           !   0 = 32 bit
                           !   1 = 64 bit
```

OUTPUT PARAMETERS:

```

integer      rc      ! Error return code:
              !   rc = 0    all is well
              !   rc = -12   error determining default precision
              ! NetCDF Errors
              !
              ! -----
              !   rc = -36   error from ncaptc/ncapt (global attribute)
              !   rc = -55   error from ncredf (enter define mode)
              !   rc = -56   error from ncdef (exit define mode)

```

REVISION HISTORY:

1998.07.30	Lucchesi	Initial interface design.
1998.07.30	Lucchesi	Initial coding.
1998.09.24	Lucchesi	Changed error handling.
1998.09.28	Lucchesi	Added support for multiple precisions

CONTENTS:

```

real*4 dummy32
real*8 dummy64
real r
integer i
real*4, allocatable :: buf32(:)
real*8, allocatable :: buf64(:)
integer err

call ncredf ( fid, rc )
if (err("PutRealAtt: could not enter define mode",rc,-55) .NE. 0)
.     return

if (HUGE(dummy32) .EQ. HUGE(r) .AND. prec .EQ. 0) then      ! -r4
  call ncapt ( fid, NCGLOBAL, name, NCFLOAT, count, buf, rc ) ! 32-bit out

else if (HUGE(dummy32) .EQ. HUGE(r) .AND. prec .EQ. 1) then ! -r4
  allocate (buf64(count))                                     ! 64-bit out
  do i=1,count
    buf64(i) = buf(i)
  enddo
  call ncapt ( fid, NCGLOBAL, name, NCDOUBLE, count, buf64, rc )
  deallocate (buf64)

else if (HUGE(dummy64) .EQ. huge(r) .AND. prec .EQ. 0) then ! -r8
  allocate (buf32(count))                                     ! 32-bit out
  do i=1,count
    buf32(i) = buf(i)
  enddo

```

```

    enddo
    call ncapt ( fid, NCGLOBAL, name, NCFLOAT, count, buf32, rc )
    deallocate (buf32)

else if (HUGE(dummy64) .EQ. huge(r) .AND. prec .EQ. 1) then ! -r8
    call ncapt ( fid, NCGLOBAL, name, NCDOUBLE, count, buf, rc ) ! 64-bit out

else
    rc = -12
    return
endif
if (err("PutRealAtt: error writing attribute",rc,-36) .NE. 0)
.   return

call ncendf ( fid, rc )
if (err("PutRealAtt: could not exit define mode",rc,-56) .NE. 0)
.   return

rc = 0
return
end

!-----
!      NASA/GSFC, Data Assimilation Office, Code 910.3, GEOS/DAS      !
!-----
```

2.10 GFIO_PutCharAtt – Write a user-defined character attribute (Source File: gfo.F)

This routine allows the user to define a character (string) attribute in an open GFIO file.

INTERFACE:

```
subroutine GFIO_PutCharAtt ( fid, name, count, buf, rc )
```

USES:

```
Implicit NONE
#include "netcdf.inc"
#include "gfio.h"
```

INPUT PARAMETERS:

```
integer      fid          ! File handle
character*(*) name        ! Name of attribute
integer      count        ! Number of characters to write
character    buf(count)  ! Buffer containing string
```

OUTPUT PARAMETERS:

```

integer      rc      ! Error return code:
              !   rc = 0    all is well
              ! NetCDF Errors
              !
              ! -----
              !   rc = -36  error from ncaptc/ncapt (global attribute)
              !   rc = -55  error from ncredf (enter define mode)
              !   rc = -56  error from ncendf (exit define mode)

```

REVISION HISTORY:

1998.07.30	Lucchesi	Initial interface design.
1998.07.30	Lucchesi	Initial coding.
1998.09.24	Lucchesi	Changed error handling.

CONTENTS:

```
!-----
integer err

call ncredf ( fid, rc )
if (err("PutCharAtt: could not enter define mode",rc,-55) .NE. 0)
.   return
call ncaptc ( fid, NCGLOBAL, name, NCCHAR, count, buf, rc )
if (err("PutCharAtt: error writing attribute",rc,-36) .NE. 0)
.   return
call ncendf ( fid, rc )
if (err("PutCharAtt: could not exit define mode",rc,-56) .NE. 0)
.   return

rc = 0
return
end

!-----
```

NASA/GSFC, Data Assimilation Office, Code 910.3, GEOS/DAS

```
!-----
```

2.11 GFIO_GetAttNames – Get global attribute names (Source File: *gfio.F*)

This routine allows the user to get the names of global attributes.

INTERFACE:

```
subroutine GFIO_GetAttNames ( fid, ngatts, aname, rc )
```

USES:

```
Implicit NONE
#include "netcdf.inc"
#include "gfio.h"
```

INPUT PARAMETERS:

```
integer fid ! File handle
```

INPUT/OUTPUT PARAMETERS:

```
integer ngatts ! Expected number of attributes (input)
               ! Actual number of attributes (output if rc=-2)
```

OUTPUT PARAMETERS:

```
character(*) fname(ngatts) ! Array of attribute names
integer rc ! Error return code:
           ! rc = 0 all is well
           ! rc = -10 ngatts is incompatible with file
           ! rc = -11 character string not long enough
           ! NetCDF Errors
           ! -----
           ! rc = -48 error from ncinq
           ! rc = -57 error from ncanam
```

REVISION HISTORY:

1998.08.05	Lucchesi	Initial interface design.
1998.08.05	Lucchesi	Initial coding.
1998.09.24	Lucchesi	Changed error handling.

CONTENTS:

```
!-----
integer ngattsFile, i
integer nDims, dimSize, recDim
integer err

! Make NetCDF errors non-fatal, but issue warning messages.

call ncpopt(NCVERBOS)

! Check number of attributes against file

call ncinq (fid,nDims,dimSize,ngattsFile,recdim,rc)
if (err("GetAttNames: ncinq failed",rc,-48) .NE. 0) return
if (ngattsFile .NE. ngatts) then
  rc = -10
```

```

    ngatts = ngattsFile
    return
  endif

! Check length of fname string

  if (LEN(fname(1)) .lt. MAXNCNAME) then
    print *, 'GFIO_GetAttNames: length of fname array must be at ',
    .      'least ',MAXNCNAME,' bytes.'
    rc = -11
    return
  endif

! Read global attribute names

  do i=1,ngatts
    call ncanam (fid, NCGLOBAL, i, fname(i), rc)
    if (err("GetAttNames: error reading attribute name",rc,-57)
    .     .NE. 0) return
  enddo

  rc = 0
  return
end

!-----
!           NASA/GSFC, Data Assimilation Office, Code 910.3, GEOS/DAS      !
!-----
```

2.12 GFIO_AttInquire – Get information about an attribute (Source File: *gfio.F*)

This routine allows the user to get information about a global attribute of an open GFIO file. This is most useful for determining the number of values stored in an attribute.

INTERFACE:

```
subroutine GFIO_AttInquire ( fid, name, type, count, rc )
```

USES:

```

Implicit NONE
#include "netcdf.inc"
#include "gfio.h"
```

INPUT PARAMETERS:

```

integer          fid          ! File handle
character*(*)   name         ! Name of attribute
```

OUTPUT PARAMETERS:

```

integer type      ! Code for attribute type
                  !   0 = integer
                  !   1 = real
                  !   2 = character
                  !   3 = 64-bit real
                  !   4 = 64-bit integer
                  !   -1 = other
integer count    ! Number of items (length of array)
integer rc        ! Error return code:
                  !   rc = 0    all is well
                  !   NetCDF Errors
                  !
                  ! -----
                  !   rc = -58  error from ncainq

```

!NOTES: The returned integer "type" for 64-bit integer is not supported in the current implementation of netCDF/HDF. When a user writes a 64-bit integer attribute using PutIntAtt, it is actually saved as a 64-bit real by the HDF library. Thus, upon reading the attribute, there is no way for HDF/GFIO to distinguish it from a REAL number. The user must realize this variable is really an integer and call GetIntAtt to read it. Even for a 64-bit integer, type=4 will never be returned unless there are changed to HDF/netCDF.

REVISION HISTORY:

1998.07.30	Lucchesi	Initial interface design.
1998.07.30	Lucchesi	Initial coding.
1998.09.24	Lucchesi	Changed error codes, added type assignment.

CONTENTS:

```

integer nctype
integer err

call ncainq (fid, NCGLOBAL, name, nctype, count, rc)
if (err("AttInquire: error reading attribute info",rc,-58)
     .NE. 0) return
if (nctype .EQ. NCLONG) then
  type = 0
elseif (nctype .EQ. NCFLOAT) then
  type = 1
elseif (nctype .EQ. NCCHAR) then
  type = 2
elseif (nctype .EQ. NCDOUBLE) then
  type = 3
else

```

```

    type = -1
  endif

  rc = 0
  return
end

!-----  

!      NASA/GSFC, Data Assimilation Office, Code 910.3, GEOS/DAS      !
!-----  


```

2.13 GFIO_GetIntAtt – Read a user-defined integer attribute (Source File: *gfio.F*)

This routine allows the user to read an integer attribute from an open GFIO file.

INTERFACE:

```
subroutine GFIO_GetIntAtt ( fid, name, count, buf, rc )
```

USES:

```

Implicit NONE
#include "netcdf.inc"
#include "gfio.h"
```

INPUT PARAMETERS:

```

integer      fid          ! File handle
character(*) name        ! Name of attribute
```

INPUT/OUTPUT PARAMETERS:

```

integer      count       ! On input: Number of items in attribute
                      ! On output: If rc = -1, count will contain
                      !             the correct count of this attribute
```

OUTPUT PARAMETERS:

```

integer      buf(count) ! Buffer with integer values
integer      rc         ! Error return code:
                      !   rc = 0    all is well
                      !   rc = -1   invalid count
                      !   rc = -2   type mismatch
                      !   rc = -12  error determining default precision
                      ! NetCDF Errors
                      ! -----
                      !   rc = -36  error from ncaptc/ncapt (global attribute)
                      !   rc = -51  error from ncagtc/ncagt (global attribute)
```

REVISION HISTORY:

1998.07.30	Lucchesi	Initial interface design.
1998.07.30	Lucchesi	Initial coding.
1998.09.29	Lucchesi	Changed error handling. Added 64-bit support.

CONTENTS:

```

integer length, type
integer err, i
integer*4 dummy32
integer*8 dummy64
integer*4, allocatable :: buf32(:)
integer*8, allocatable :: buf64(:)

call ncainq (fid, NCGLOBAL, name, type, length, rc)
if (err("GetIntAtt: error reading attribute info",rc,-58)
.     .NE. 0) return

if ( count .NE. length ) then
  rc = -1
  count = length
  return
endif

if ( type .NE. NCLONG .AND. type .NE. NCDOUBLE) then
  rc = -2
  return
endif
if ( HUGE(dummy32) .EQ. HUGE(i)) then
  if ( type .EQ. NCLONG ) then          ! -i4 32bit
    call ncagt ( fid, NCGLOBAL, name, buf, rc )
  else          ! type .EQ. NCDOUBLE
    allocate (buf64(count))           ! -i4 64bit
    call ncagt ( fid, NCGLOBAL, name, buf64, rc )
    do i=1,count
      buf(i) = buf64(i)
    enddo
    deallocate (buf64)
  endif
else if (HUGE(dummy64) .EQ. HUGE(i)) then
  if ( type .EQ. NCLONG ) then
    allocate (buf32(count))          ! -i8 32bit
    call ncagt ( fid, NCGLOBAL, name, buf32, rc )
    do i=1,count
      buf(i) = buf32(i)
    enddo
  endif
endif

```

```

      deallocate (buf32)
      else           ! type .EQ. NCDOUBLE
      call ncagt ( fid, NCGLOBAL, name, buf, rc ) ! -i8 64bit
      endif
   else
      rc = -12
      return
   endif
   if (err("GetIntAtt: error reading attribute value",rc,-51)
.     .NE. 0) return

   rc = 0
   return
end

!-----
!      NASA/GSFC, Data Assimilation Office, Code 910.3, GEOS/DAS      !
!-----
```

2.14 GFIO_GetRealAtt – Read a user-defined real attribute (Source File: *gfio.F*)

This routine allows the user to read a real attribute from an open GFIO file.

INTERFACE:

```
subroutine GFIO_GetRealAtt ( fid, name, count, buf, rc )
```

USES:

```

Implicit NONE
#include "netcdf.inc"
#include "gfio.h"
```

INPUT PARAMETERS:

```

integer      fid      ! File handle
character(*) name      ! Name of attribute
```

INPUT/OUTPUT PARAMETERS:

```

integer      count      ! On input: Number of items in attribute
                      ! On output: If rc = -1, count will contain
                      !             the correct number of attributes
```

OUTPUT PARAMETERS:

```

real      buf(count) ! Buffer with real values
integer  rc          ! Error return code:
```

```

      !   rc = 0    all is well
      !   rc = -1   invalid count
      !   rc = -2   type mismatch
      !   rc = -12  error determining default precision
      ! NetCDF Errors
      !
      ! -----
      !   rc = -36  error from ncaptc/ncapt (global attribute)
      !   rc = -51  error from ncagtc/ncagt (global attribute)

```

REVISION HISTORY:

1998.07.30	Lucchesi	Initial interface design.
1998.07.30	Lucchesi	Initial coding.
1998.09.29	Lucchesi	Changed error handling. Added 64-bit support.

CONTENTS:

```

integer length, type
integer err
real r
integer i
real*4 dummy32
real*8 dummy64
real*4, allocatable :: buf32(:)
real*8, allocatable :: buf64(:)

call ncainq (fid, NCGLOBAL, name, type, length, rc)
if (err("GetRealAtt: error reading attribute info",rc,-58)
     .NE. 0) return

if ( count .NE. length ) then
  rc = -1
  count = length
  return
endif
if ( type .NE. NCFLOAT .OR. type .NE. NCDOUBLE) then
  rc = -2
  return
endif

if ( HUGE(dummy32) .EQ. HUGE(r)) then
  if ( type .EQ. NCFLOAT ) then          ! -r4 32bit
    call ncagt ( fid, NCGLOBAL, name, buf, rc )
  else                                ! type .EQ. NCDOUBLE
    allocate (buf64(count))           ! -r4 64bit
    call ncagt ( fid, NCGLOBAL, name, buf64, rc )

```

```

        do i=1,count
          buf(i) = buf64(i)
        enddo
        deallocate (buf64)
      endif
    else if (HUGE(dummy64) .EQ. HUGE(r)) then
      if ( type .EQ. NCFLOAT ) then
        allocate (buf32(count))           ! -r8 32bit
        call ncagt ( fid, NCGLOBAL, name, buf32, rc )
        do i=1,count
          buf(i) = buf32(i)
        enddo
        deallocate (buf32)
      else           ! type .EQ. NCDOUBLE
        call ncagt ( fid, NCGLOBAL, name, buf, rc ) ! -r8 64bit
      endif
    else
      rc = -12
      return
    endif
    if (err("GetRealAtt: error reading attribute value",rc,-51)
      .NE. 0) return

    rc = 0
    return
  end

!---
!       NASA/GSFC, Data Assimilation Office, Code 910.3, GEOS/DAS      !
!---

```

2.15 GFIO_GetCharAtt – Read a user-defined character attribute (Source File: *gfio.F*)

This routine allows the user to read a character attribute from an open GFIO file.

INTERFACE:

```
subroutine GFIO_GetCharAtt ( fid, name, count, buf, rc )
```

USES:

```

Implicit NONE
#include "netcdf.inc"
#include "gfio.h"
```

INPUT PARAMETERS:

```
integer fid ! File handle
character*(*) name ! Name of attribute
```

INPUT/OUTPUT PARAMETERS:

```
integer count ! On input: Number of items in attribute
! On output: If rc = -1, count will contain
! the correct number of attributes
```

OUTPUT PARAMETERS:

```
character buf(count) ! Buffer with character values
integer rc ! Error return code:
! rc = 0 all is well
! rc = -1 invalid count
! rc = -2 type mismatch
! NetCDF Errors
!
! -----
! rc = -36 error from ncaptc/ncapt (global attribute)
! rc = -51 error from ncagtc/ncagt (global attribute)
```

REVISION HISTORY:

1998.07.30	Lucchesi	Initial interface design.
1998.07.30	Lucchesi	Initial coding.
1998.09.29	Lucchesi	Changed error handling.

CONTENTS:

```
integer length, type
integer err

call ncainq (fid, NCGLOBAL, name, type, length, rc)
if (err("GetCharAtt: error reading attribute info",rc,-58)
.     .NE. 0) return
if ( count .NE. length ) then
  rc = -1
  count = length
  return
endif
if ( type .NE. NCCHAR) then
  rc = -2
  return
endif

call ncagtc ( fid, NCGLOBAL, name, buf, count, rc )
if (err("GetCharAtt: error reading attribute value",rc,-51)
.     .NE. 0) return
```

```
rc = 0
return
end
```